SYSTEM IDENTIFICATION FOR NONLINEAR CONTROL USING NEURAL NETWORKS

Robert F. Stengel[1] and Dennis J. Linse[2]
Department of Mechanical and Aerospace Engineering
Princeton University
Princeton, NJ

ABSTRACT

An approach to incorporating artificial neural networks in nonlinear, adaptive control systems is described. The controller contains three principal elements: a nonlinear-inverse-dynamic control law whose coefficients depend on a comprehensive model of the plant, a neural network that models system dynamics, and a state estimator whose outputs drive the control law and train the neural network. Attention is focused on the system-identification task, which combines an extended Kalman filter with generalized-spline function approximation (e.g., basis splines, back-propagation feedforward networks, or cerebellar model articulation controller networks). Continual learning is possible during normal operation (without taking the system off line for specialized training). Nonlinear-inverse-dynamic control requires smooth derivatives as well as function estimates, imposing stringent goals on the approximating technique.

INTRODUCTION

Current research on artificial neural networks has been spurred by the need for pattern recognition, optimization, and control algorithms that can learn from experience. The biological paradigm provides an attractive starting point, as learning is the crux of living activity. Nevertheless, it remains to be seen whether or not biologically-motivated algorithms (i.e., artificial neural networks) are computationally appealing, particularly as compared to more conventional numerical techniques for function approximation. These techniques also can "learn" or "be taught," although those terms rarely are used for the process of fitting curves or determining coefficients from empirical data.

The most important question appears to be how well neural networks generalize multivariate functions of high input dimension. Basis splines (or B-splines) represent an efficient, modern, conventional technique [1, 2]. A univariate B-spline function approximation can be trained quickly and accurately using deterministic or averaging techniques, and the resulting spline approximant has nice numerical properties, such as positivity and compact support. Multivariate B-splines can be computed, but they may become cumbersome in both training and operation if the number of independent variables is much larger than two. Furthermore, the multivariate B-spline requires a fairly rigid definition of the input space, resulting in potentially high computation times and storage requirements. Neural networks can, in principle, gain efficiency through automatically identifying and concentrating on significant receptive regions in the input space [3, 4].

Neural networks can be incorporated in nonlinear control logic in at least two ways. The first is as a *generalized feedforward control function* that maps the space of desired command responses into corresponding control settings. This neural network is conceptually straightforward, as it is based on the inverse of the plant's response to control inputs [5]. The second is to use the neural network as a *model of plant dynamics*, which is embodied in the control law but is distinct from the command/control specification [as in nonlinear-inverse-dynamic (NID) control]. The first approach may lead to more compact implementation, while the second may provide more flexibility for multi-modal or fault-tolerant control.

In the remainder of the paper, we discuss elements of the second approach. After briefly stating the nonlinear control problem and its NID solution, a learning structure for neural networks based on an extended Kalman filter is described, and details of two network formulations are presented. It is shown that smooth neural network outputs are desirable and, in fact, required for NID control. The back-propagation feedforward network has sufficient smoothness as normally defined, but the cerebellar model articulation controller network must be modified to eliminate quantization effects.

THE NONLINEAR CONTROL PROBLEM

Consider the control of a nonlinear dynamic plant as described by,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x}, \mathbf{u}) + \mathbf{w} \tag{1}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{2}$$

$$\mathbf{z} = \mathbf{h}_z(\mathbf{x}) + \mathbf{n} \tag{3}$$

with $\mathbf{x}$ and $\mathbf{w} \in R^n$, $\mathbf{u}, \mathbf{y} \in R^m$, and $\mathbf{z}$ and $\mathbf{w} \in R^l$. $\mathbf{f}$ and $\mathbf{g}$ are smooth vector fields in R^n; $\mathbf{h}$ and $\mathbf{h}_z$ are smooth vectors fields in R^m and R^l. $\mathbf{x}$ represents the dynamic state, $\mathbf{u}$ is its control, $\mathbf{w}$ is the disturbance input, $\mathbf{y}$ is the response vector, $\mathbf{z}$ is the measurement of the process, and $\mathbf{n}$ is the measurement error.

For illustration, assume that eq. 1 and 2 can be written as,

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\,\mathbf{u} + \mathbf{w} \tag{4}$$

---

[1]Professor
[2]Graduate Research Assistant

Presented at the 1990 Conference on Information Sciences and Systems, Princeton, NJ, March, 1990.

$$y = H x \tag{5}$$

with $w = 0$ and $z = x$. Define the *derivative response vector* $y^{(d)}$ by termwise differentiation of eq. 5 and substitution of eq. 4, repeated until the control affects each element of $y^{(d)}$ linearly [6, 7]:

$$y^{(d)} = f^*(x) + G^*(x)\, u \tag{6}$$

$G^*(x)$ is guaranteed to be nonsingular, so it is possible to construct an *integrator-decoupling control law* of the form,

$$u = c(x) + C(x)v \tag{7}$$

where,

$$C(x) = [G^*(x)]^{-1} \tag{8}$$

$$c(x) = -[G^*(x)]^{-1} f^*(x) \tag{9}$$

and v is a command input vector. Together with additional compensation to provide satisfactory closed-loop stability and command response, eq. 7 forms a *nonlinear-inverse-dynamic control law*.

For typical response vector specifications in aircraft control applications [8, 9], one to three differentiations are required to bring out the control effect in $y^{(d)}$; hence, up to two partial derivatives of $f(x)$ and $G(x)$ (with respect to x) must exist in those cases. Our goal is to approximate an uncertain but observable portion of the system model using neural networks. Consequently, neural network outputs must be smooth enough for the differentiation and inversion contained in eqs. 6, 8, and 9 to take place.

THE SYSTEM IDENTIFICATION PROBLEM

Consider a system described by eq. 1-3 with $w = 0$ and $z = x$, in which $f(x)$ is known without error and $g(x,u)$ is subject to uncertainty or change. Let $g(t)$ represent the value of $g(x,u)$ at time t during some period of system operation; then $g(t)$ could be estimated precisely as

$$g(t) = \dot{x}(t) - f[x(t)] \tag{10}$$

and $\{g(t), x(t), u(t)\}$ would form a suitable training set for a functional approximation to $g(x,u)$. During the course of normal system operation, it is assumed that $x(t)$ and $u(t)$ would provide a succession of training sets that were sufficiently rich for function approximation, and it would be possible to "replay" system trajectories off line for improved learning. There is no restriction on the form of $g(x,u)$; it is suggested here that $g(x,u)$ be represented by one or more *neural networks*.

More realistically, w and n are not zero, and z is not x, so eq. 10 cannot be evaluated. However, with the usual observability requirements, it is possible to make estimates of $g(t)$ and $x(t)$ using an *Extended Kalman Filter*, as described, for example, in [10]. Following a procedure analogous to *Estimation Before Modeling* [11-15], the state vector is augmented to include $g(t)$ and enough derivatives of $g(t)$ to assure a smooth estimate of $g(t)$. [In the off-line aircraft system identification applications presented in [11-15] (Fig. 1), $g(t)$ was treated as an integrated random-walk (Gauss-Markov) process, necessitating the estimation of $\dot{g}(t)$ and $\ddot{g}(t)$ as well as $g(t)$. Improved estimates were obtained by following the filtering step with a reverse-time modified Bryson-Frazier (MBF) smoothing step [16].] In the current application, the filtered estimates $\hat{g}(t)$ and $\hat{x}(t)$ are used with $u(t)$ to train the neural network $\hat{g}(\hat{x}, u)$. Smoothed estimates could be used for further off-line training improvement. [See [17] for a related application of filtering and smoothing].

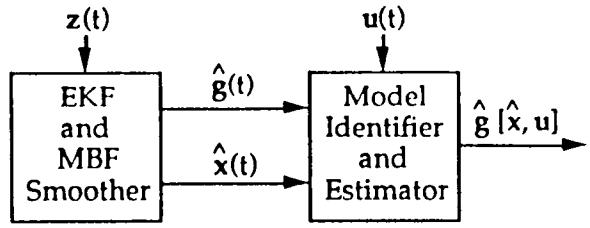

Figure 1. Basic Estimation Before Modeling.

For the airplane example, $x(t)$ is an 11-component vector representing 3 components (each) of translational and rotational rates, 4 quaternion components, and altitude. $f[x(t)]$ represents kinematic and inertial effects, while the vector $g(t)$ has just 6 nonzero terms due to aerodynamic, thrust, and control effects. These terms are axial, side, and normal specific forces [$X(t)$, $Y(t)$, and $Z(t)$] and roll, pitch, and yaw specific moments [$L(t)$, $M(t)$, and $N(t)$]. Consequently, 6 parallel neural networks would be formed for this application. Aerodynamic effects can be normalized using air density, airspeed, reference area, and reference length; hence, the neural networks would represent the non-dimensional coefficients $C_X(x,u)$, $C_Y(x,u)$, $C_Z(x,u)$, $C_l(x,u)$, $C_m(x,u)$, and $C_n(x,u)$.

The neural network can be integrated with NID control, as shown in Fig. 2. The dashed arrow from the estimation model block to the nonlinear control law block represents an update of the functional form of the control law rather than the direct feedback path of the other connections.

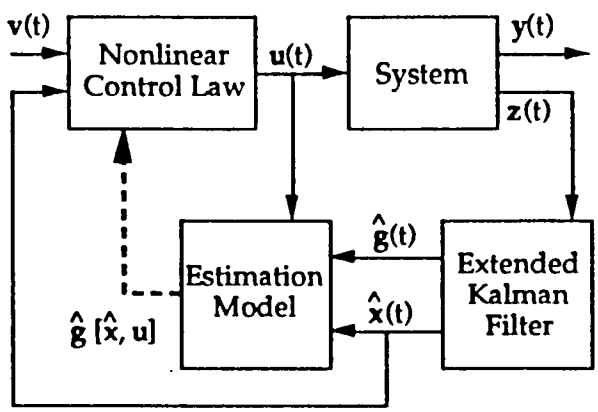

Figure 2. Integration of Neural-Network System Identification with Nonlinear-Inverse-Dynamic Control.

NEURAL NETWORK ESTIMATION SCHEMES

Back-propagation feedforward neural networks and CMAC neural networks are particularly suitable for system identification. Each represents a different structural model of a process and presents unique capabilities for the system identification process. Although neural network techniques appear very different from classic techniques such as B-splines, under close examination there are strong similarities. As discussed in [18], most neural networks, including the two models discussed here, can be viewed as generalized splines. This observation opens the door on a large body of knowledge about function approximation. A brief description of the two networks follows. Further details about each can be found in [19].

Back-Propagation Feedforward Network

One of the simplest, and most used, artificial neural network is the back-propagation feedforward network [20]. A feedforward neural network is constructed from a weighted interconnection of simple nonlinear elements called *nodes*. The nodes are separated into *layers*, with the input to each node exclusively from the previous layer and the output connected only to the following layer. The orderly construction leads to a simple recursive definition

$$t^{(k)} = W^{(k-1)} x^{(k-1)} \tag{11}$$

$$x^{(k)} = s^{(k)}\left[t^{(k)}\right] \tag{12}$$

for layers k = 1 to N. The *activation function* vector, $s^{(k)}[t]$, is formed by concatenating the scalar *activation functions* contained in each node of a single layer:

$$s^{(k)}[t] = [\sigma_1(t_1) \ldots \sigma_n(t_n)]^T \tag{13}$$

One input to each layer is fixed at unity and serves as a threshold, as illustrated by Fig. 3.

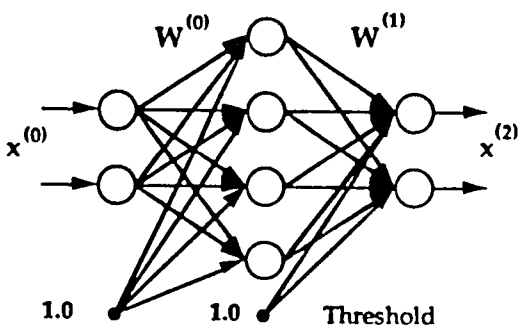

Figure 3. A Simple Feedforward Neural Network.

Given the form of the activation function (usually the same in each node), the number of layers, and the number of nodes in each layer, the *back-propagation rule* [20] can be used to update the values of the weights, $W^{(k)}$. The back-propagation rule attempts to minimize the mean-square error of the difference between the network output and a desired output value using a gradient-based descent algorithm. The choice of activation functions and numbers of layers and nodes is the subject of much research [e.g. [21]].

Cerebellar Model Articulation Controller

The Cerebellar Model Articulation Controller (CMAC) neural network [3, 4] is a generalization of a simple table look-up of a multi-input/single-output nonlinear function. Originally modeled on the uniform structure of the cerebellum, the basic CMAC divides the multi-dimensional input space into a series of overlapping hypercubes. A weight is assigned to each hypercube. The output of the CMAC for a specific input point is simply the sum of the weights of those hypercubes enclosing the input point. Applications to date have been in the areas of function approximation [3, 4, 17], robotics [5,22,23], and adaptive control [24].

Introducing an *association space*, $\mathcal{A}$, the operation of the CMAC is best described by two mappings. The first goes from the input space, X, to the association space

$$Q: \; X \rightarrow \mathcal{A} = \{0, 1\}^{N_A} \tag{14}$$

The elements of the binary *selector vector*, **a**, in the N_A dimensional $\mathcal{A}$ are fixed by

$$a_j = \begin{cases} 1 & x \in \Lambda_j \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where Λ_j is the j^{th} receptive region. The *receptive regions* are the overlapping hypercubes that cover the input space. By overlapping the receptive regions such that any input lies within exactly C receptive regions, **a** has C non-zero values. The size of C controls the generalization capability of the CMAC.

The second mapping from **a** to the scalar output, y,

$$\mathcal{R}: \; \mathcal{A} \rightarrow \mathcal{Y} \tag{16}$$

is simply a summation of the weights associated with the selected receptive regions. Defining a weight vector, **w**, of dimension N_A, the inner product of **w** with **a** generates the desired output

$$y = \mathbf{w}^T\mathbf{a} \tag{17}$$

The basic CMAC (Fig. 4) quantizes the inputs based on the size of the overlap of the receptive regions. Since there is only one further linear operation performed, the output of the CMAC is a piecewise-continuous function.

Since the weights and output are connected by a simple linear operation, a learning algorithm is easy to prescribe. Each weight contributing to a particular output value is adjusted by a fraction of the difference between the network output and the desired output. The fraction added is determined by both the speed of desired learning and the number of weights contributing to the output.

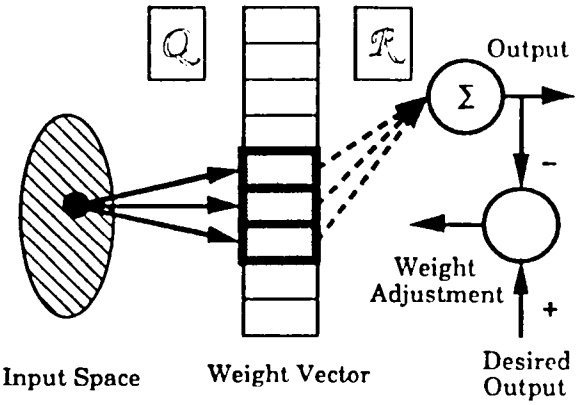

Figure 4. The Basic CMAC Architecture.

By its construction, the fidelity of the learned CMAC response is directly related to the quantization of the input space into receptive regions. For multi-dimensional input spaces, the number of receptive regions, and therefore the number of weights, scales as

$$N_A = \vartheta(Q^n) \tag{18}$$

where Q is the number of number of divisions in each input dimension and n is the number of input dimensions. For even reasonably sized, multi-input problems, the number of weights quickly overwhelms the memory capacity of most computers. Fortunately, in many applications, such as control, only small regions of (or trajectories through) the input space are used. By implementing a many-to-few mapping of the weight vector using a random hash-coding technique [25], the number of necessary weights can be compressed to a reasonable size. For example, in [23], a CMAC was applied to a robotics problem with 15 inputs. The weight vector was compressed using a random hash function from the original large length down to lengths between 32,768 and 8 weights. Virtually no degradation in performance was visible for lengths down to 1024.

DIFFERENTIABILITY OF THE ESTIMATION SCHEMES

The application to NID control imposes strong restrictions on the estimation scheme to be used, including the the existence of smooth derivatives. The two neural network techniques described have differing capabilities in this regard.

Recognizing that the overall function of the feedforward network is

$$x^{(N)} = g\left[x^{(0)}\right] \tag{19}$$

the derivative of the network function, $\partial g / \partial x^{(0)}$, can be calculated recursively as

$$\frac{\partial x^{(k)}}{\partial x^{(k-1)}} = \frac{\partial s^{(k)}[t^{(k)}]}{\partial t^{(k)}} W^{(k-1)} \tag{20}$$

Since the weight matrices are constant, the differentiability of the network is determined by the activation function used in each node. One of the most common activation functions is the sigmoid

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad \forall\, z \in R \tag{21}$$

Since this function is infinitely differentiable, the full network also is infinitely differentiable.

The basic CMAC network is very different from the feedforward network. The quantization of the input space causes the CMAC to have piecewise-continuous output as demonstrated in Fig. 5 for a coarse quantization of the input value.

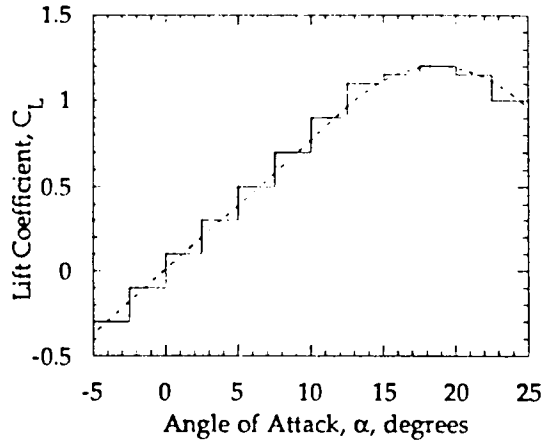

Figure 5. Example CMAC output for very coarse receptive regions.

Although useful in many other situations, the basic CMAC cannot be used in NID control because the control laws require continuous derivatives. Noticing the similarity between the receptive regions of CMACs and first-order basis splines, Lane et al. introduce *higher-order CMACs* in [26]. Equation 15 is redefined to be

$$a_j = \begin{cases} B_{n,j}(x) & x \in \Lambda_j \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

where the basis functions, B, of order, n, differ from conventional B-splines [basis functions] [1, 2, 26] to account for differing support (or receptive) regions. Due to the similarity of the basis functions, higher-order CMACs inherit the continuity properties of B-splines, including (n - 2) continuous derivatives. The penalty paid for this increase in continuity is the increased computation necessary in eq. 22 when compared to eq. 15. An example of receptive regions for a higher-order CMAC is given in Fig. 6.

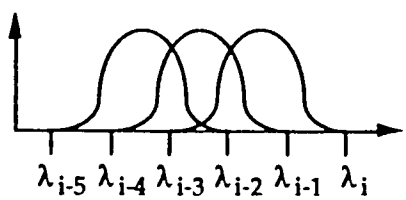

Figure 6. Receptive Regions of a Higher-Order CMAC.

EXAMPLE NEURAL NETWORK SYSTEM IDENTIFICATION

As a preliminary step to the full implementation of the integrated neural-network, NID control law, a simple investigation of the capabilities of the neural networks was performed. A one-dimensional test function representing lift coefficient variation with angle-of-attack for the airplane example is given in Fig. 7.

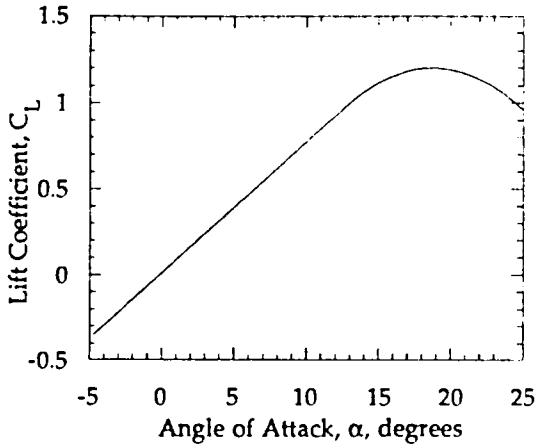

Figure 7. Lift curve test function

The learning capabilities of neural networks were tested by presenting random angles-of-attack between -5° and 25°, and the corresponding lift coefficients to each network. The result after 50,000 random training samples presented to a back-propagation network with two hidden layers of 10 and 5 nodes respectively, is given in Fig. 8. The error between the network output and the actual lift curve is very small, indicating a good representation of the original function.

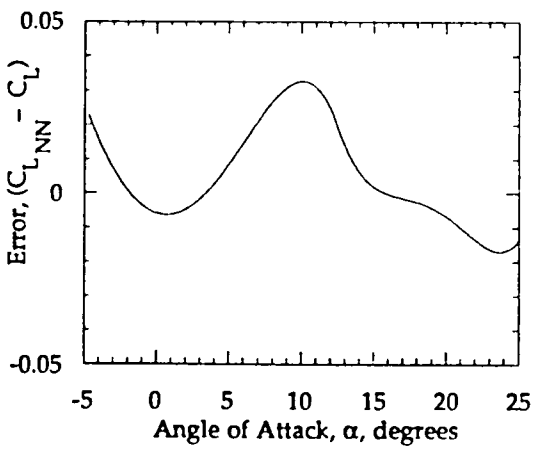

Figure 8. Back-propagation error for lift curve test function

The basic CMAC network network is similarly tested by presenting random input/output pairs. The result after 2000 such presentations for a network with 100 weights and a generalization value, C, of 10 (Fig. 9) also is very good. The noisy appearance of the network error is due to the piecewise continuity of the CMAC output. Higher-order CMACs will alleviate this problem.

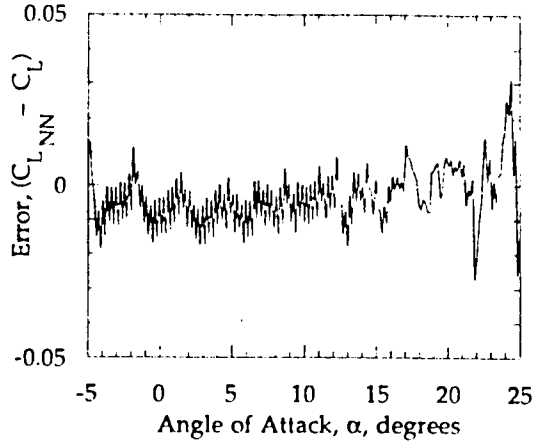

Figure 9. CMAC error for lift curve test function

Further testing of the neural networks, including multi-dimensional inputs and higher-order CMACs, remains to be completed. Full implementation of the integrated neural-network, NID control law (Fig. 2) will follow.

CONCLUSIONS

Neural networks provide a feasible alternative for function approximation in adaptive (or learning) nonlinear control systems. Parallel neural networks representing generalized forces due to state and control can be trained using the outputs of an extended Kalman filter. Neural network outputs must be continuously differentiable for use with nonlinear-inverse-dynamic control logic. Back-propagation feedforward neural networks using sigmoidal activation functions have the necessary differentiability. Cerebellar model articulation controller neural networks must be modified to achieve the required smoothness. Determining the practicality of neural-network/nonlinear-inverse-dynamic controllers is an important area for future research.

ACKNOWLEDGMENTS

This research has been sponsored by the FAA and the NASA Langley Research Center under Grant No. NGL 31-001-252 and by the Army Research Office under Grant No. DAAL03-89-K-0092.

REFERENCES

1. M.G. Cox, "Practical Spline Approximation", in P.R. Turner, Ed., *Topics in Numerical Analysis, Lecture Notes in Mathematics*, Vol. 965, Springer-Verlag, Berlin, 1982, pp. 79-112.

2. S.H. Lane, "Theory and Development of Adaptive Flight Control Systems Using Nonlinear Inverse Dynamics," Ph.D. Dissertation, Princeton University, June, 1988.

3. J.S. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 97, September 1975, pp. 220-227.

4. J.S. Albus, "Data Storage in the Cerebellar Model Articulation Controller (CMAC)," *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 97, September 1975, pp. 228-233.

5. W.T. Miller, R.P. Hewes, F.H. Glanz, and L.G. Kraft, "Real-Time Dynamic Control of an Industrial Manipulator Using a Neural-Network-Based Learning Controller," *IEEE Transactions on Robotics and Automation*, Vol. 6, No. 1, February, 1990, pp. 1-9.

6. S.N. Singh and W.J. Rugh, "Decoupling in a Class of Nonlinear Systems by State Variable Feedback," *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, Series G, Vol. 94, 1972, pp. 323-329.

7. A. Isidori, *Nonlinear Control Systems: An Introduction*, Springer-Verlag, Berlin, 1989.

8. S.N. Singh and A. Schy, "Output Feedback Nonlinear Decoupled Control Synthesis and Observer Design for Maneuvering Aircraft," *International Journal of Control*, Vol. 31, No. 4, 1980, pp. 781-806.

9. S.H. Lane and R.F. Stengel, "Flight Control Design Using Nonlinear Inverse Dynamics," *Automatica*, Vol. 24, No. 4, July, 1988, pp. 471-484.

10. R.F. Stengel, *Stochastic Optimal Control, Theory and Application*, John Wiley & Sons, Inc., New York, 1986.

11. H.L. Stalford, "High-Alpha Aerodynamic Model Identification of T-2C Aircraft Using the EBM Method," *Journal of Aircraft*, Vol. 18, 1981, pp. 801-809.

12. C. Fratter and R.F. Stengel, "Identification of Aerodynamic Coefficients Using Flight Testing Data," *AIAA Atmospheric Flight Mechanics Conference*, AIAA-83-2099, August, 1983.

13. M. Sri-Jayantha and R.F. Stengel, "Determination of Nonlinear Aerodynamic Coefficients Using the Estimation-Before-Modeling Method", *Journal of Aircraft*, Vol. 25, No. 9, September, 1988, pp. 796-804.

14. M. Sri-Jayantha and R.F. Stengel, "Data Acquisition System for High Angle of Attack Parameter Estimation," *Technical Soaring*, Volume IX, No. 4, pp. 85-95.

15. X. Silhouette and R.F. Stengel, "Estimation of the Aerodynamic Coefficients of the Navion Aircraft at High Angles of Attack and Sideslip," *AIAA Atmospheric Flight Mechanics Conference*, AIAA-87-2622, August, 1987, pp. 452-463.

16. G.J. Bierman, "Fixed Interval Smoothing with Discrete Measurements," *International Journal of Control*, Vol. 18, No. 1, January, 1973, pp. 65-75.

17. G. Rodriguez, "Kalman Filtering, Smoothing, and Recursive Robot Arm Forward and Inverse Dynamics," *Journal of Robotics and Automation*, Vol. RA-3, No. 6, December, 1987, pp. 624-639.

18. T. Poggio and F. Girosi, "Regularization Algorithms for Learning That Are Equivalent to Multilayer Networks," *Science*, Vol. 247, February 23, 1990, pp. 978-982.

19. D.J. Linse and R.F. Stengel, "Neural Networks for Function Approximation in Nonlinear Control," to be presented at the 1990 American Control Conference, San Diego, CA, May, 1990.

20. D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognitions, Volume 1: Foundations*, D.E. Rumelhart and J.L. McClelland, Eds., MIT Press, Cambridge, Massachusetts, 1986.

21. G. Cybenko, "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, Vol. 2, No. 4, 1989, pp. 303-314.

22. D.A. Handelman, S.H. Lane, and J.J. Gelfand, "Characteristics of Integrated Rule-Based Systems and Neural Networks Enabling Robotic Skill Acquisition," *Proceedings of the BIRA Seminar on Neural Networks*, Ghent, Belgium, April 1989.

23. W.T. Miller, F.H. Glanz, and L.G. Kraft, "Application of a General Learning Algorithm to the Control of Robotic Manipulators," *International Journal of Robotics Research*, Vol. 6, No. 2, 1987, pp. 84-98.

24. L.G. Kraft and D.P. Campagna, "A Comparison of CMAC Neural Network and Traditional Adaptive Control," *Proceedings of the 1989 American Control Conference*, Vol. 1, 1989, pp. 884-889.

25. D.E. Knuth, *The Art of Computer Programming*, Vol. 3, Addison-Wesley Publishing Company, Reading, MA, 1973.

26. S.H. Lane, D.A. Handelman, and J.J. Gelfand, "Development of Adaptive B-Splines Using CMAC Neural Networks," *1989 IEEE INNS International Joint Conference on Neural Networks (IJCNN)*, Vol. 1, Washington, D.C., July 1989, pp. 683-688.